一种基于指纹融合的跨语言剽窃检测技术 *

刘　刚，左　权，杨倩茹

(哈尔滨工程大学 计算机科学与技术学院, 哈尔滨 150001)

摘　要：跨语言剽窃一直是学术不端现象发生的重灾区，也是极难发现的一种剽窃行为。跨语言剽窃的检测和识别技术是目前最亟待发展的技术，也是反剽窃抄袭领域的最大技术难点。在总结和分析了单语剽窃检测和跨语言剽窃检测国内外研究现状的基础上，针对跨语言剽窃检测存在的问题，提出了一种基于指纹融合的跨语言剽窃检测技术，并将所提出的技术在人工构建的剽窃集上进行实验验证，对实验结果进行详细分析和对比分析，验证了该技术的有效性。

关键词：中间指纹；指纹融合；语义消歧；跨语言剽窃检测

中图分类号：TP391　　doi: 10.3969/j.issn.1001-3695.2017.07.0672

Cross-language plagiarism detection technology based on fingerprint fusion

Liu Gang, Zuo Quan, Yang Qianru

(*College of Computer Science & Technology*, *Harbin Engineering University*, *Harbin* 150001, *China*)

Abstract: Cross-language plagiarism has always been the hardest hit for academic misbehavior. It is also a behavior that is extremely difficult to spot. Cross-language plagiarism detection and identification technology is the most urgent technology that needed to be developed. It is also the biggest technical difficulty in the field of plagiarism. Based on the summary and analysis of current researches on the monolingual plagiarism detection and cross-language plagiarism detection, aiming at the existing problem of cross-language plagiarism detection, this paper proposes a cross-language plagiarism detection technology based on fingerprint fusion. This paper also carries out experimental verification on the plagiarism set of artificial building. Through analyzing and comparing the result of experiments, it can be concluded that the method is indeed effective.

Key Words: intermediate fingerprint; fingerprint fusion; semantic disambiguation; cross-language plagiarism detection

0　引言

随着互联网技术的快速发展，人们可以通过各种途径方便的获取各类信息。然而，网络便利生活的同时也使得任何人都可以轻易的拷贝别人的内容作为自己的内容，把别人的既有观点当成自己的创新观点，这样在无意中就形成了剽窃的行为。Alzahrani 等人将剽窃的类型分为两类，一类是字面剽窃，即剽窃者不需要花费太多时间隐藏他们的学术犯罪行为，只是进行简单的复制和粘贴文本；另一类是智能剽窃，即剽窃者试图通过把别人的贡献改变成自己的来欺骗读者，一般用翻译、替换同义词等各种智能的方式来试图隐藏、混淆和改变原来的工作[1~3]。而在其中跨语言剽窃形成的文档，在语言形式上基本没有可比性，通过软件自动检测很困难，目前这方面的研究开展的比较少，并且还没有公开实用的软件问世。McCabe 的一项研究表明，在 18000 名学生中，40%的学生承认他们至少剽窃过一次，其中包括跨语言剽窃[4]。不仅国外，国内也有类似的事情发生，从中学生到博士，从最简单的照搬照抄到替换同义词、移位变换，学术剽窃的现象层出不穷。剽窃检测不仅是对知识产权的一种维护，更是对知识工作者的尊重。剽窃检测已经成为一个研究领域，尤其是在学术领域，这是为了减少侵权现象的发生以及研究剽窃行为的类型。研究跨语言剽窃检测技术在实际系统中的应用，不仅对遏制学术不端行为具有重大意义，还能够保障原始作者的权益，有利于学术的进步。

1　研究现状

国外对于跨语言剽窃检测的研究也才刚刚兴起，并处在快速发展阶段。2008 年，Alberto 等人[5]使用统计模型进行跨语言剽窃检测，该研究依赖于由平行语料库产生的统计双语词典，并且依赖于双语对齐算法。2008 年，Ceska 等人[6]提出一种基于字词所在位置的跨语言剽窃检测方法 MLPlag。该方法采用 EuroWordNet 将单词转换成一种语言的独立表示，而且作者建立了两个多语言语料库：JRC-EU 和童话故事。2007 年，Potthsh 等人[7]引入了一个新的多语言检索模型——跨语言显示语义分析，用来分析跨语言的相似性，其分别在多语言平行语料库

基金项目：黑龙江省博士后科研启动金资助项目（LBH-Q15031）；黑龙江省教育科学规划课题（GJC1215107）

作者简介：刘刚（1976-），男，副教授，主要研究方向为智能信息处理、数据库与知识库、社会网络（liugang@hrbeu.edu.cn）；左权（1993-），男，硕士，主要研究方向为自然语言处理、知识图谱；杨倩茹（1992-），女，硕士，主要研究方向为自然语言处理、文字血缘.

(JRC-Acquis)和多语言可比较语料库(Wikipedia)上进行了实验。2010 年，他们又将此方法和其他方法进行了比较，发现使用字符 n-gram 能达到一个更好的效果，然而，基于字符 N-Grams 的方法并不适用于语法无关对的语言之间的比较[8]。2009 年，Pinto 等人[9]为了解决基于机器翻译的跨语言剽窃中翻译和剽窃检测总是作为两个单独步骤进行处理的问题，以及其中累计的不精确等问题，提出了直接把这两步整合为一步的方法，以提高检测的精确度。2010 年，Pereirad 等人[10]提出了一种跨语言剽窃检测的新方法。该方法分为五个主要阶段；语言规范化、候选文档的检索、分类器训练、剽窃行为分析和后期处理。2013 年，Alberto 等人[11]提出一个免费的可使用的跨语言剽窃检测框架，他们在这个框架上探索了三种跨语言相似性评估模型的适用性：基于对齐的跨语言相似性分析(CL-ASA)，基于 n-grams 的跨语言剽窃检测和基于机器翻译的剽窃检测(T+MA)。同年，Marc Franco-Salvador 等人[12]提出了一种基于多语言的语义网络进行跨语言剽窃检测的方法。该方法使用的多语言语义网络为 BabelNet，它是一个有向图，其节点表示多语言的概念和命名实体，边表示它们之间的语义关系。为了验证其有效性，他们在 2011 年 PAN 中德语-英语和西班牙语-英语的剽窃语料上进行实验，实验结果表明，基于多语言的语义网络使语言独立起来，在剽窃检测方面优于其他方法。2014 年，Aljohani 等人[13]提出使用 Winnowing 算法对阿拉伯语和英语之间的跨语言剽窃进行检测。该算法在维基百科上进行，使用精准率和召回率作为评价标准。

跨语言剽窃检测在国外是刚刚起步，目前正处于快速发展阶段，而国内对这方面的研究很少。国外有阿拉伯语-英语，德语-英语等等，但由于中文的特殊性，国外的跨语言剽窃检测技术有些并不适用于国内。由于词语往往具有一词多义的现象，和中文相比较语言差异尤其明显，所以从英文文献中直接翻译过来形成中文这种抄袭很难检测出来。针对这个问题，本文针对跨语言剽窃检测技术进行了研究，以达到跨语言文本剽窃检测的目的。

2　潜在剽窃文档集生成

2.1　数字指纹的相关概念

数字指纹是把文本中的某些特征通过某种选取策略进行 Hash 计算而生成的数字编码[14]。如果直接对原文本进行字符串的匹配会存在很多问题，比如，存储空间大、效率低、精度不够。因此需要把文本映射成指纹进行剽窃检测。

为了评价文本剽窃程度，需要计算两个文本指纹的相似度，因为文本所对应的指纹应该能够很好的表示该文本。根据数字指纹的相关定义得知，生成指纹时需要考虑以下几点：文本块粒度、指纹的选取策略、文本数量以及函数的选择问题[15]。

文本粒度是指用于生成数字指纹的文本长度。文本粒度的选择最后会对剽窃结果的精度产生很大的影响。最大的指纹粒度是整个文本，这样只能检测出原封不动复制粘贴的文本，对稍有变动的剽窃检测不出来；最小的是一个字符，这样容易导致生成的指纹过多，效率过慢，而且会产生很多错误的匹配，使精度下降。指纹的选取策略有全指纹选取、基于频率的选取、基于结构的选取和基于位置的选取。而关于文本块的选择问题直接和指纹数量相关，指纹数量太多准确度高，但计算量和存储空间大了。因此需要选择合适的指纹数量进行计算。

2.2　基于 WordNet 的中间指纹编码

WordNet 是由 Princeton 大学建立的一种基于认知语言学的机器可读词典。WordNet 描述的对象包括英语复合词(compound)、短语动词(phrasal verb)、搭配词(collocation)、成语(idiomatic)和单词(word)。其中，最基本的单位是单词，WordNet 中没有比词更小的结构单位，也没有比词更大的组织单位。它既和传统的词典不同，也和同义词典不同，它是混合了这两种类型词典特征的词典。指纹的建立过程就是把选取的字符串映射为唯一整数的过程。由于名词的对应关系在各种语言中最为明确，一篇文章的大部分内容也为名词，而名词在 WordNet 中有最为清晰的树型结构，因此本算法只对名词进行指纹编码。WordNet 中的名词是以树型结构组织的，所有的名词都在一棵以(entity.n.01)为根的树的下面。语义关系是 WordNet 中最重要的关系，词义关系其是最基本构件之一，而词义在 WordNet 中是以同义词集来表示的。WordNet 中的树型结构是以同义词集为节点的，每个节点代表一个语义或是概念。WordNet 中的上下位关系就是父节点和子节点的关系，父节点即上位词(hypernyms)，一般比该节点表示的意思更抽象，即从根节点到叶节点词义越来越具体，越来越专业。图 1 显示了 WordNet2.1 中名词的树型结构的一部分。

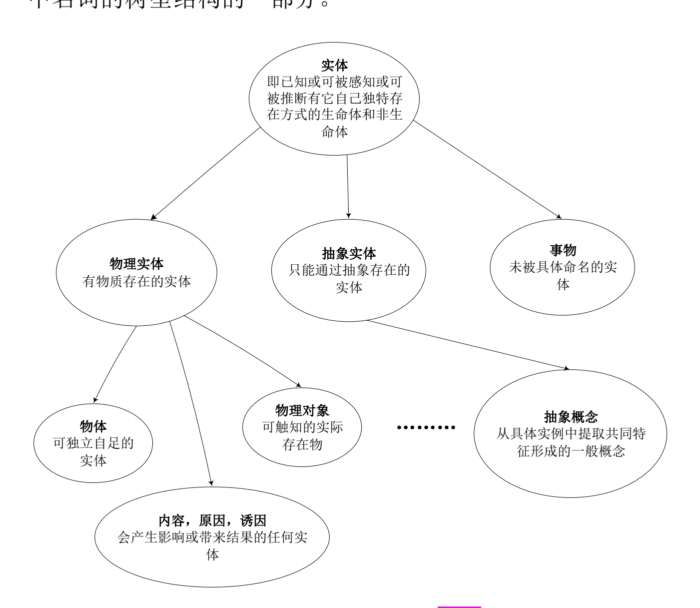


图 1　WordNet 中名词同义词集树型结构示例

由于很多语言都有与英文 WordNet 对应的版本，这就在一定程度上跨越了语言的屏障，因为当将不同语言的 WordNet 对应到一起时，每个同义词集已经是一个与语言无关的语义节点。只有通过某种自然语言才能恰当的表示这些语义。在 WordNet3.0 版本中同义词集共有 117659 个，其中名词的同义

词集为 82115 个，占了同义词集的 80%左右。本文将所有的名词同义词集进行指纹编码，其产生的指纹就是独立于语言的一个语义中间层，所以称为中间指纹。考虑到后续需要对名词进行语义消歧以及指纹提取，同时也为了提高效率，算法 1 使用以下思想对 WordNet 中的名词同义词集进行指纹编码：

a）子节点的编码以父节点的编码为前缀；

b）用 $levelbit_i$ 位二进制编码第 i 层，其中 $levelbit_i = levelm_i + 1$，$levelm_i$ 是第 i 层的最大子节点数；

c）从最高位开始编码，用 1 到 $levelbit_1$ 位二进制编码第一层，用 $levelbit_1$ 到 $levelbit_1 + levelbit_2$ 位二进制编码第二层，以此类推。

图 2 是中间指纹编码算法的一个图解。

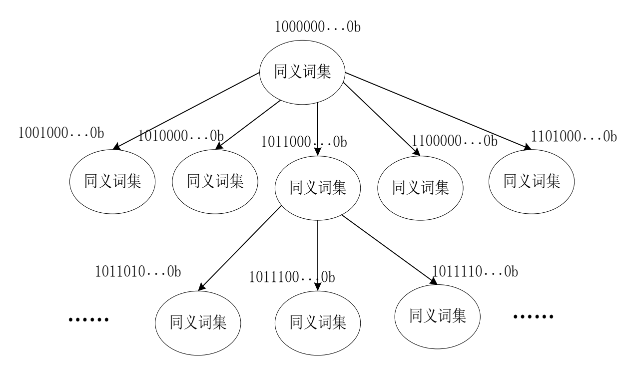


图 2　中间指纹编码示例图

中间指纹编码算法如下：

算法 1　中间指纹编码

输入：预处理完的文本；

输出：WordNet 中的同义词集经过编码后的中间指纹 。

定义一个空的队列 $queue$；

往队列里添加 WordNet 名词的树型结构的根节点；

定义一个字典 $dict = \{queue[0]:[1]\}$；

往字典的第一项对应的 $value$ 中添加 $\mathrm{len}(queue[0].hyponynms)$ // $hyponynms$ 是下位词；

往字典的第一项对应的 $value$ 中再添加 1 对应的二进制形式；

每层编码所需的二进制位数 $levelbit = [1,2,4,6,9,7,8,9,9,9,9,7,7,5,5,6,5,4,4,1]$

$while$（$queue$ 的长度>0）{

$count = 1$；

$node = queue.pop(0)$；

$for\ node.hyponynms()$ 中的每一个子节点 $child$ {

将 $child$ 添加至队列 $queue$；

层数 $level$ 为 $dict[node][0]$ 加 1；

level 层对应的编码长度 $codeLen$ 为 $levelbit[level-1]$；

$level$ 层对应的编码 $codeLen$ 为 $dict[node][2]$ 加上 $codeLen$ 长的 $count$ 的二进制表示；

$dict[child] = [level, len(child.hyponynms(), codestr$

}

$count = count + 1$；

}

$for\ dict$ 中的每一项 x

将 $dict[x][2]$ 左对齐，扩展为 117 位，不足的位用 0 补齐

2.3　文本预处理

由于是在名词的基础上进行的散列，需要对中英文文本进行预处理，提取出本文所需要的名词。

采用 ICTCLAS 对中文文本进行分词和词性标注，然后提取出其中的名词存放在列表里。中科院的汉语词法分析系统 ICTCLAS 在多年的研究工作积累上，对分词的准确率达到 95%以上，并且在国内 973 专家组组织的测评中获得第一名。它是基于层叠的隐马尔可夫模型(cascaded hidden Markov model, CHMM)的。本文采用斯坦福自然语言处理小组开发的词性标注工具 Stanford log-linear part-of-speech tagger 对英文文本进行词性标注和词干化。该工具通过一个双向依赖网络对英文文本进行标记，并且在多个连续的单词之间考虑了词汇关系，有效地利用了线性模型中的先验条件，词性标注的准确率达到了 97.24%。

2.4　基于中间指纹的语义消歧

经过文本预处理之后，可以得到名词序列，由于存在一词多义的现象，需要确定这个词在上下文中的意思。虽然有些文本预处理考虑到了语义消歧，但是只能在单语的情况下知道词在上下文的意思，对于跨语言则无可奈何。本文采用基于中间指纹的与语义无关的消歧算法对名词序列进行消歧——主要是利用概念相关性原理在消歧窗口中包含的所有词的义项中选取多个，通过计算语义密度来进行消歧。消歧的结果是语义密度最大的子树包含的义项。假设消歧窗口的大小是 19，窗口中都是提取出的名词，而中间的词就是被消歧的词，比如：$\{r_1, r_2, L, r_9, r_{10}, r_{11}, L, r_{18}, r_{19}\}$，$r_{10}$ 是被消歧的词，每次确定一个词的义项后，窗口向后移动一个，此时 r_{11} 为被消歧的词，以此类推，直到所有的名词都确定义项。对于窗口长度为 $2l+1$ 的 R：$\{r_a, r_{a+1}, L, r_{a+l-1}, r_{a+l}, r_{a+l+1}, L, r_{a+2l-1}, r_{a+2l}\}$，被消歧词是 r_{a+l}，消歧算法的主要步骤如下：

a) 将包含 R 中每个 r_i 的同义词集 $Synset(r_i) = \{r_{i_1}, r_{i_2}, L\}$ 合并为一个大的候选集 $C = \bigcup_{i=a}^{a+2l} synset(r_i)$；

b)对候选集 C 中的所有同义词集按照它们对应的中间指纹排序；

c)计算 C 中任意几个同义词集的语义密度，r_{a+l} 的消歧结果就是语义密度最大的子树下的同义词集；

d)往后移动一个窗口，重复上述步骤，直到所有名词都被消歧。

基于中间指纹的语义消歧算如下：

算法 2　基于中间指纹的语义消歧

输入：文本段落提取出的名词序列 $listnon$；

输出：经过消歧之后所确定的义项的指纹序列 $senseen$。

定义变量 $a=0$， $l=9$；

将序列 *senseen* 置为空;

while（a 小于名词序列 *listnon* 的长度）{

　设置临时变量最大语义密度 $MaxD=0$；

　设置临时列表 R 为空;

　将 $\left\{\begin{matrix} r_a, r_{a+1}, \cdots, r_{a+l-1}, r_{a+l}, r_{a+l+1}, \\ \cdots, r_{a+2l-1}, r_{a+2l} \end{matrix}\right\}$ 即名词序列的第 a 到第 $a+2l$ 元素赋值给列表 R；

　设置候选集 $C=\bigcup_{i=a}^{a+2l} synset(r_i)$；

　对候选集 C 按照中间指纹进行排序;

　设置临时变量 p 为候选集 C 的长度;

　$for\ (i=0; i<p; i++)\{$

　　$for\ (j=i+1; j\le p; j++)\{$

　　　设置 $C'=\{r_i, r_{i+1}, \cdots, r_j\}$；

　　　$if\ C'$ 和 $Synset(r_{a+l})$ 的交集的数目等于 1 或者等于 2{

　　　　将 $d(C')$ 的语义密度赋值给 D；

　　　　if（D 大于最大语义密度 $MaxD$）{

　　　　　$MaxD=D$；

　　　　　将 C' 和 $Synset(r_{a+l})$ 的交集赋值给 T；

　　　　}// if 结束

　　　}// if 结束

　　}//内层 for 循环结束

　}//外层 for 循环结束

　将 T 添加进序列 *senseen*；

　$a=a+1$；

}// *while* 循环结束

返回指纹序列 *senseen*

2.5　指纹选取

指纹的选取对后续相似度的检测和剽窃检测的精准度有直接的影响，正确的指纹选取能充分的表示文档本身，反之，不正确的指纹选取与原文档会有很大的偏差。全指纹的选取是最简单的，同时在检测准确度和性能上较其他选取策略较高，但对于大文件，效率就有了明显的下降。基于位置的选取依赖于指纹的位置，对于打乱顺序的剽窃，准确度不高。基于结构的指纹选取考虑的是论文的篇章结构，对于智能的剽窃和语言结构不一样的剽窃检测效果不理想。基于频率的指纹选取就是依据频率来选取指纹，过滤一些宽泛的、频率过高的指纹，而选取一些频率较合适的指纹。由于本文是提取出文本的名词作为特征，但是有的名词很常用，不具有代表性，因此需要过滤掉，选取合适的名词指作为文档的指纹。

本文采用了一个变通的方法来解决这个问题。对于 WordNet 中的名词同义词集树型结构，上层节点的语义比下层宽泛，下层节点的语义比上层具体。采用同义词集在树型结构中的深度作为过滤特征集的条件。这是因为在各种语言中，人们使用各个名词的语义的总体频率是大致一样的，只是具体表现的形式不同而已。还因为各种名词的语义在 WordNet 树型结构的深度越小通常具有越高的全局频率[16]。这里的全局频率是在指某个语义在语言中出现的频率，可以当做是在无穷大的语料库上训练得到的。并且根据平均语义和深度的关系，前四层全局频率随深度增加而增加，从第五层到第二十层全局频率随深度增加而减少，由于深度小的节点语义宽泛区分能力不强，因此本文把这些特征值过滤掉，即把低 100 位全为 0 的指纹过滤掉，剩下的就是文档所对应的指纹。

具体指纹选取算法如下：

算法 3　指纹选取

输入：中文文本 D，英文文本 D'；

输出：中文指纹 $finger1$，英文指纹 $finger2$。

对中文文本 D 进行段落划分，形成小的文档 d，并存储在 S_1；

对英文文本 D' 进行段落划分，形成小的文档 d'，并存储在 S_2；

for 段落集合 S_1 中的每一个段落 d {

　对 d 进行分词和词性标注，存储在序列 *listch* 中;

　for listch 中的每一个项{

　　if 该项的词性是名词

　　　添加进序列 *listnonch* 中;

　}// *for* 循环结束

}// *for* 循环结束

for 段落集合 S_2 中的每一个段落 d' {

　对 d' 进行分词和词性标注，并进行词干化，存储在序列 *listen* 中;

　for listen 中的每一个项{

　　if 该项的词性是名词

　　　添加进序列 *listnonen* 中;

　}// *for* 循环结束

}// *for* 循环结束

调用算法 2 对 *listnonch*， *listnonen* 进行消歧;

for listnonch 每一项{

　if 该项的低 100 位不全为 0

　　就把该添加进 $finger1$；

}// *for* 循环结束

对于 *listnonen* 也是执行一样的操作，最后形成 $finger2$；

返回中文指纹 $finger1$，英文指纹 $finger2$；

跨语言剽窃一般使用开源的翻译软件，对文本进行翻译后，粘贴到自己的论文中。而对跨语言剽窃进行检测时不可能对所有的源文本都进行详细的分析，因此需要先对源文本进行检索，检索出可能是剽窃的段落之后，再进行详细分析。

至此已经介绍了跨语言剽窃检测的前一部分，潜在剽窃文档集的提取，其中最核心的是跨语言文本相似度的计算。图 3 描述了跨语言指纹相似度计算的全部流程，这里以中英文为例。

3　剽窃检测结果详细认定

3.1　SimWin 指纹融合算法

跨语言剽窃一般是翻译过后稍加修改而形成的，如何更加确定为剽窃，需要更加细化的分析和计算。由于语言之间的结

构和表述不同，本文无法精确到词或者像单语言剽窃检测一样，精确到十几个字或者连续的词组，而一般剽窃会是以句子为单位进行翻译，从而进行跨语言剽窃。因此本文以句子作为最小单元进行跨语言剽窃检测的详细分析。

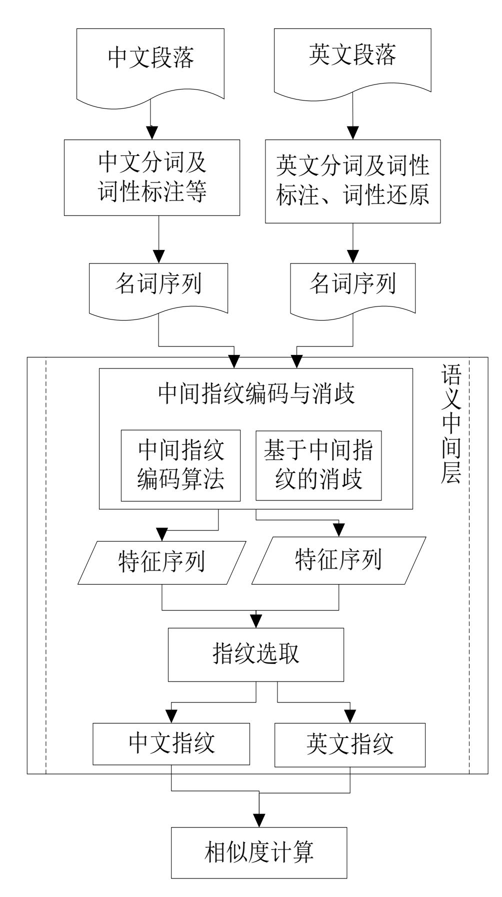


图 3　语言指纹相似度计算流程图

SimHash 算法[17]是一种局部敏感的指纹算法。传统的指纹算法不能通过指纹衡量出两个文本的相似程度，而 SimHash 算法可以通过指纹之间的汉明距离来衡量两个文本的相似程度。SimHash 算法常用于海量的文本相似度计算，由于 SimHash 的核心思想是降低维度，并且把一个高维的特征向量哈希成一个固定长度的指纹，从而通过比较指纹计算文本的相似度。但是其效率对应的代价是精度的下降。Winnowing 算法的去噪功能很好，通过滑动窗口的方式来提取文本的特征序列。它的优点是在文本有小的变动时，即哈希序列有稍微的改变时，所提取的文本特征序列基本不变，而且在文本有小的变动时，窗口大小对最后的检测结果基本没有影响，这样就增强了算法的鲁棒性。但是其有特征指纹数量太多、选取指纹不全等缺点。本文综合考虑 SimHash 算法和 Winnowing 算法的优缺点，以及两种算法提取指纹的侧重点也有所不同，把两种算法融合在一起，提出了 SimWin 算法，以便更准确的来计算句子之间的相似度。本文采用如下公式进行指纹融合。

$$S(A,B)=\alpha*(1-\frac{H(A,B)}{f})+(1-\alpha)*S_{winnowing}(A,B)$$

其中：$H(A,B)$ 是句子 A 和句子 B 的汉明距离，f 是 SimHash 算法中产生指纹的位数，$S_{winowing}(A,B)$ 是句子 A 和句子 B 通过 Winnowing 算法计算得出的相似度，α 是 SimHash 算法的权重，$1-\alpha$ 是 Winnowing 算法结果的权重，具体的取多少可以更加准确的衡量两个句子的相似度，需要通过实验来决定。

根据上述描述，SimWin 算法具体步骤如下：首先对所有文本进行分句；接着按照算法计算句子之间的汉明距离，按照 Winnowing 算法计算相应两个句子之间的相似度；最后按照指纹融合公式，计算两个句子之间的最终相似度。具体指纹融合算法如下所示：

算法 4　SimWin 指纹融合

输入：文本段落 D，文本段落 D'；

输出：经过指纹融合后两两句子之间的相似度 Sim。

对文本段落 D 进行分句，形成句子序列 S_1；

对文本段落 D' 进行分句，形成句子序列 S_2；

for 句子序列 S_1 中的每一项 S {

　调用 SimHash 算法形成其指 $f_{simhash}$ 纹；

　使用 Winnowing 算法形成其指纹 $f_{winnowing}$；

　$for\ S_2$ 中的每一个句子 S' {

　　调用 SimHash 算法形成其指纹 $f'_{simhash}$；

　　使用 Winnowing 算法形成其指纹 $f'_{winnowing}$；

　　计算 $f_{simhash}$ 和 $f'_{simhash}$ 之间的汉明距离；

　　应用公式 $R=\frac{F(A)\cap F(B)}{F(A)\cup F(B)}$ 计算 $f_{winnowing}$ 和 $f'_{winnowing}$ 之间的相似度；

　　应用 $S(A,B)$ 公式计算 s 和 s' 融合后的相似度 r；

　　将 r 存入 Sim 中；

　}//内层 for 循环结束

}//外层 for 循环结束

对 Sim 进行排序；

返回 Sim；

相对于只用单一的指纹算法来计算句子之间的相似度，本文提出的基于指纹融合的句子相似度计算方法综合了两种指纹算法的特性，将两者最终的结果很好的融合在一起，使得最终的结果更加准确，也提高了鲁棒性。图 4 是剽窃检测详细分析过程图。

3.2　剽窃片段合并

通过句子之间的相似度计算以及通过阈值的过滤，可以得到可疑文本中的句子是否剽窃了源文档中的句子。之所以采用句子作为剽窃检测的基本单位，是因为语言与语言之间的结构顺序的不同，没办法像单语一样进行确定到字的剽窃检测，而句子作为一个基本的单元，可以作为最小的检测单位。但是会出现连续剽窃的两个句子，面对这种情况，剽窃检测最终的结果中应该只出现一个检测结果而不是两个，如图 5 所示。

图 5 表示文档 A 的分析结果，分析方法检测到可疑文本两个剽窃句子。这两个句子都是从文本 B 剽窃而来的。第一个剽窃在可疑文本中从第 1000 个字符的位置开始，剽窃了 500 个字符，相应的源文本 B 中从第 3000 个字符的位置开始，也有

500 个字符的长度。从图 5 可以看出，第二次检测表明，有一个剽窃正好在前一次检测出的剽窃句子后面，因此，需要把这两个检测连成一个，而不是报告两个剽窃。

文本合并后的结果如图 6 所示。

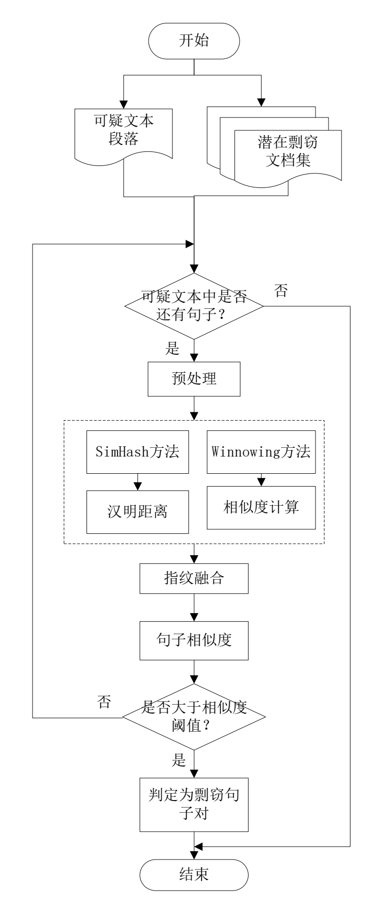


图 4　剽窃检测详细分析过程图


```
<document reference="A.txt">
<feature name="detected-plagiarism"
        this_offset="1000" this_length="500"
        source_reference="B.txt"
        source_offset="3000" source_length="500"
/>
<feature name="detected-plagiarism"
        this_offset="1500" this_length="300"
        source_reference="B.txt"
        source_offset="3500" source_length="300"
/>
</doucument>
```


图 5　合并之前的剽窃结果图


```
<document reference="A.txt">
<feature name="detected-plagiarism"
        this_offset="1000" this_length="800"
        source_reference="B.txt"
        source_offset="3000" source_length="800"
/>
</doucument>
```


图 6　合并之后的剽窃结果图

为了合并连续的剽窃句子，本文使用以下方法：

a)通过把源文本按照属性 source_reference 分类，从而进行集体检测；

b)对于 a)中得到的每一个集合，将它们按照属性 this_offset 的大小按照升序排序；

c)把最多相距一个预定义的字符数的相邻的检测连接起来；

d)对于每一个剽窃段落只报告一个剽窃检测结果(选取源文档中最大长度的段落)，即保证可疑文本中的同一段落有不超过一个可能的剽窃来源。

按照上述步骤进行剽窃结果合并，使得最终的结果整合到一起，而不是分散的剽窃检测的结果。

4　实验结果分析

跨语言剽窃检测实验主要以简体中文和英文为实验对象，由于没有很好的中英文跨语言剽窃语料库，本文通过从中国知网上下载的硕博士论文中英文摘要作为基础语料，还选取了 10 篇英文论文通过 Google 翻译成中文，随后通过人工添加、删除等操作形成本文实验所使用的剽窃语料。其中，中文文本 5000 篇，英文文本 5000 篇，存储格式都是(*.txt)。通过在这 10000 篇文本上进行实验获得相关的实验数据，并在此基础上对实验结果进行分析。本文的实验主要分为三个阶段，分别为：

a)详细分析 WordNet 的名词树型结构，通过中间指纹编码算法把名词树状结构中的同义词节点编码成指纹，通过自然语言处理技术，对中英文文本进行预处理提取其名词序列。

b)基于中间指纹对名词序列进行语义消歧，通过指纹选取策略提取出段落的中英文指纹，利用 Dice 系数对中英文指纹进行相似度计算，在中英文相似度计算的结果中通过阈值选取出潜在剽窃段落。

c)通过 Google API 把中文文本翻译成其对应的英文文本，接着对文本进行分句，然后按照 SinWin 算法计算句子之间的相似度，通过阈值选取剽窃句子，最后通过剽窃片段合并形成最后的剽窃检测结果。通过语义消歧和指纹选取之后，已经形成中英文文本所对应的指纹，然后利用 Dice 系数计算中英文文本之间的相似度。

4.1　中间指纹编码

本文实验所使用的英文 WordNet 为 PrincetonWordNet 3.0，使用的中文 WordNet 为与其对应的由南阳理工大学建立的汉语开放词网(Chinese Open Wordnet)[18]，这两者是对齐的，本文对 WordNet 的名词同义集进行编码后，以四元组的形式进行存储。四元组形式为{同义词集；英语词汇；中文词汇；中间指纹}。图 7 是部分中间指纹编码的结果。

在图 7 中，Synset(‘message.n.01’)代表同义词集，message.n.02 是单词 message 作为名词的第二个义项，其随后一行为这个义项中包含的英语词汇，在接着一行为这个义项包含的中文词汇，如果没有，则为 None，最后一行为此同义词集经过中间指纹编码算法之后对应的中间指纹，为 117 位二进制数。从图 7 可以看出，每个同义词集代表一个义项，这个义项中可能包含多个英语词汇和多个中文词汇，也可能没有相对应的中文词汇，通过计算 85%都有对应的词汇，因此对后续的实验基本没有影响。

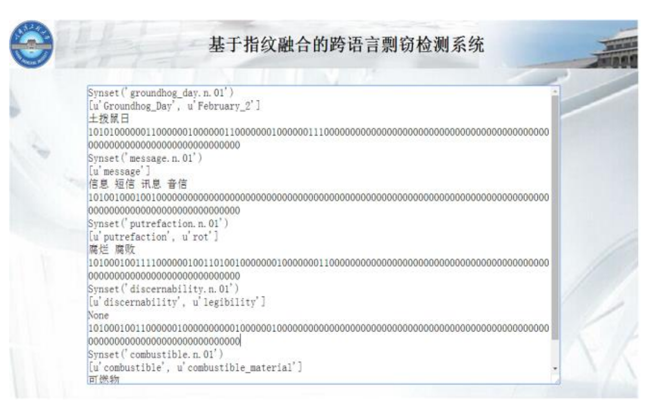


图 7　部分中间指纹编码的结果

4.2　潜在剽窃文档的检索

通过语义消歧和指纹选取之后，已经形成中英文文本所对应的指纹，然后利用 Dice 系数计算中英文文本之间的相似度。表 1 是部分中英文文本相似度计算结果。

表 1　中英文文本相似度计算部分结果

中文文本	英文文本	相似度
zh00001	en00001	0.8185642135454
zh00001	en00002	0.7802139037433
zh00001	en00003	0.7606060606061
zh00001	en00004	0.5961538461538
zh00001	en00005	0.659763313609
zh00001	en00006	0.306896551724
zh00001	en00007	0.651515151515
zh00001	en00008	0.4131578947368
zh00001	en00009	0.2920245398773
zh00001	en00010	0.104347826087
zh00001	en00011	0.154457785588
zh00001	en00012	0.3283018867925
zh00001	en00013	0.104166666667
zh00001	en00014	0.5430107526882
zh00001	en00015	0.44882478962

从表 1 可以看出“zh00001.txt”和“en00001.txt”的相似度最高。通过人工分析这两个文本，发现“zh00001.txt”和“en00001.txt”确实互为译文，这大体上验证了中英文文本相似度计算的合理性。同时为了更好的评价此算法的优劣，采用了准确率(P)、召回率(R)和 F-测度值来评测。这里的准确率和召回率的计算公式如下所示：

$$P = \frac{T}{T+P} \times 100\%$$

$$R = \frac{T}{T+N} \times 100\%$$

上述公式是在给定的相似度的阈值的情况下。其中，T 为检测为相似实际也是相似的文本数量，P 为预测为相似实际上不相似的文本数量，T 为检测为不相似但实际相似的文本的数量。F-测度值是精度和召回率的综合，$F-测度值 = 2PR/(P+R)$。图 8 是准确率、召回率和 F-测度值在不同相似度阈值下的平均值。

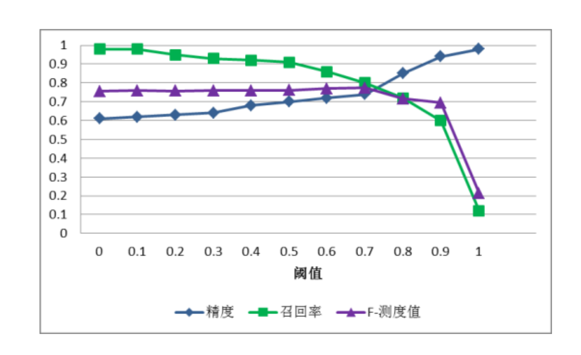


图 8　精度、召回率和 F-测度值在不同相似度阈值下的平均值

从图 8 可以看出在相似度阈值取 0.71 时，F-测度值为最佳，相似度取 0.71 也综合考虑了精度和召回率。在相似度阈值取 0.71 时，F-测度值为 0.7214，说明了中英文相似度计算的有效性。又由于此计算过程只抽取了名词作为特征序列，而且需要考虑上下文，这对较长的文本来说是可行的，但对于较短的段落和句子，该算法的精度就下降了，所以需要后续的详细分析。

4.3　SimWin 算法实验分析

在详细分析过程中，需要先把可疑文本，也就是中文文本翻译成英文，需要通过 Google API 翻译成源文档的语言（这里是英语），然后利用单语之间的相似度进行详细分析。为了更加详细的比较，本文的指纹融合算法是在句子的基础上进行的，以句号作为句子分割的标志。本文从剽窃集中取了 50 个平均句子长度在 30 个单词的句子作为测试集。计算出每个句子在不同阈值下的准确率和召回率，随后把每个阈值的平均准确率和召回率计算出来。图 9 是 SimHash 算法阈值和精度、召回率的关系图。

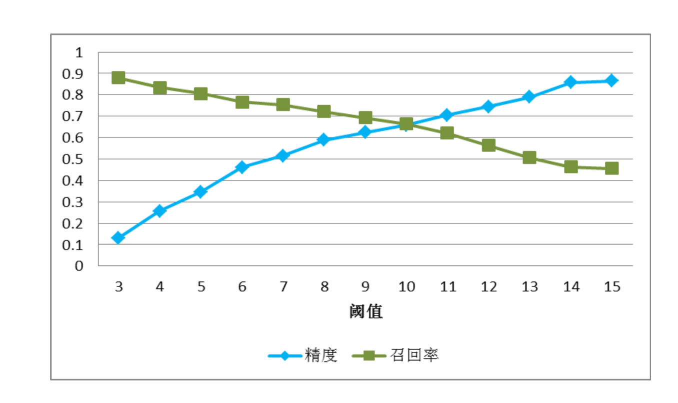


图 9　SimHash 算法阈值、精度、召回率关系图

从图 9 中可以看出当阈值为 10 时，精度和召回率达到一个平衡点，因此本文将阈值定为 10。

本文将两种算法融合到一起来衡量句子之间的相似度，但是公式中的 α 需要根据实验来确定，因为在不同的实验环境下，在实验结果最优的情况下，α 的取值不一样。本次实验和 SimHash、Winnowing 采用的测试集一样，实验选用 F-测度值

来确定α的最优值和相似度阈值。图 10 是α值、相似度阈值和 F-测度值的关系图。

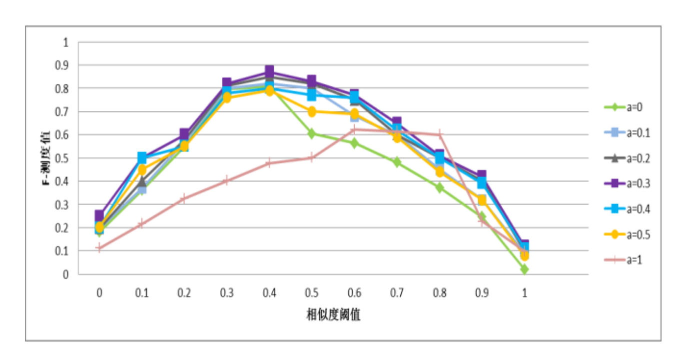


图 10 α 值、相似度阈值以及 F-测度值的关系图

从图 10 可以看出当$\alpha = 0.3$时，F-测度值要普遍高于其他值，在$\alpha = 0.3$时，相似度阈值取 0.4 时，F-测度值最高。

表 2 是其中一个句子和抽取的另外十个句子的 SimHash 汉明距离和相似度、Winnowing 相似度以及融合相似度($\alpha = 0.3$)。

表 2 句子相似度计算部分结果

英文句子	汉明距离	SimHash 相似度	Winnowing 相似度	融合相似度
sen00005	11	0.65625	0.443181	0.507103
sen00011	18	0.4375	0.050847	0.166842
sen00020	12	0.625	0.404494	0.470645
sen00055	11	0.65625	0.044642	0.241517
sen00075	15	0.53125	0.025423	0.177171
sen00086	5	0.84375	0.397849	0.531619
sen00107	18	0.4375	0.050847	0.166842
sen00108	13	0.59375	0.02309	0.194288
sen00129	10	0.6875	0.156241	0.315618
sen00310	6	0.8125	0.55	0.62875

从表 2 可以看出，当汉明距离以 10 为阈值时，只检测到两个可能剽窃的句子，而在 Winnowing 算法中，以相似度 0.35 作为阈值的话，检测到四个剽窃的句子，实际可疑句子和英文句子“sen00005”、“sen00020”、“sen00086”、“sen00310”互为译文，而一般的跨语言剽窃都是翻译而来的。所以可以说可疑句子和这四个句子互为剽窃句子。从表中可以看出 Winnowing 算法的精度要高于 SimHash 算法，但是仔细分析这十个句子，发现在 Winnowing 算法中，某些稍微类似的计算相似度太低，不利于剽窃检测，而经过融合之后就好多了，能更好的应用于剽窃检测。

图 11 是 SimHash 算法在阈值取 10、Winnowing 算法在相似度阈值取 0.35、指纹融合相似度阈值取 0.4 的精度、召回率、F-测度值的对比图。

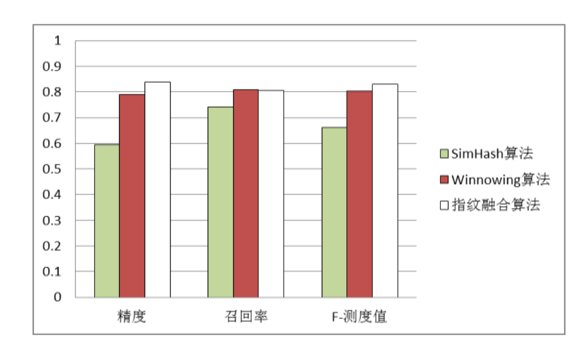


图 11 三种句子相似度计算方法结果对比图

从图 11 可以看出本文所提出的指纹融合算法在准确率和 F-测度值之间要好于 SimHash 算法和 Winnowing 算法，召回率基本和 Winnowing 算法持平，因此文本提出的指纹融合算法可认为是有效的。

在文献[11]中 Alberto 等人在同样的实验环境和语料下，比较了 CL-ASA、CL-CNG 和 T+MA 三种方法的精度和召回率，结果证明三种方法中 T+MA 的效果最好，即机器翻译加单语言剽窃分析效果最佳。其中机器翻译使用的是 Google API，翻译成同种语言之后，使用 TF-IDF 值来标记文本术语的权重，在词袋模型上使用余弦值来比较文本。这些实验都是在英语和德语以及西班牙语之间进行的，很少有以中文作为跨语言剽窃中的一种语言来对所提出的方法进行验证和评价的。本文分别对 T+MA 方法、CL-ASA 方法和基于指纹融合的跨语言剽窃检测方法进行了对比分析。图 12 是这三种方法的精度和召回率结果对比图。

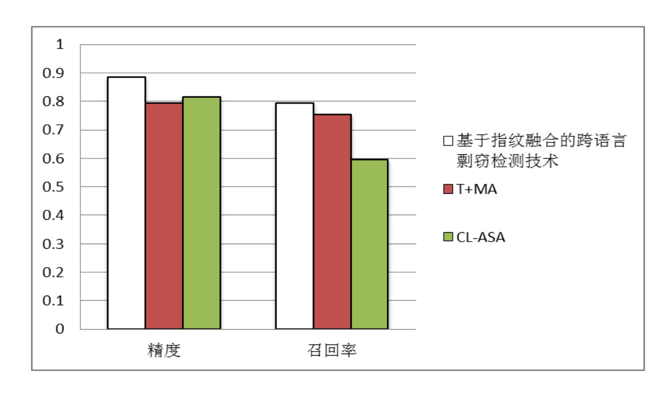


图 12 三种方法剽窃检测结果对比图

图 12 可以看出在本文的实验环境下，不管是精度还是召回率都要高于其他两种方法，精度达到 0.87，召回率达到 0.78。这验证了本文提出的基于指纹融合的跨语言剽窃检测技术的有效性。

同时由于实验都在使用指纹，而指纹的优点在于效率高，因此本文也对不同大小的剽窃集对比了检测所需的时间。图 13 是三种方法的在不同数量剽窃集上所需的时间对比图。

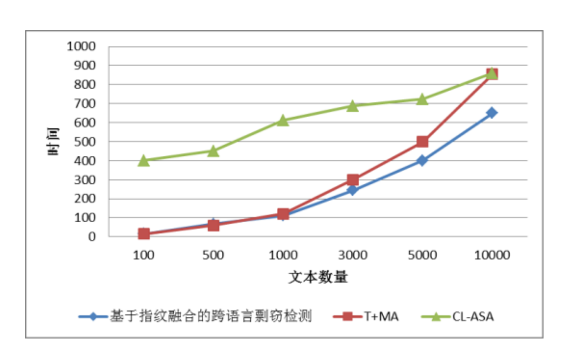


图 13 三种方法运行时间对比图

从图 13 可以明显看出在文本数量较小时本文的方法和 T+MA 的方法的时间没太大差别，但当文本增大时，本文方法所用的时间明显小于其他两种。CL-ASA 由于是先生成统计翻译模型后再进行相似度的计算，所以开始的时间会比较短。由于在文本相似度计算时采用的是位运算，后续的消歧和指纹选取也是在其基础上进行的，所以效率大大提升，而且由于是基于语义的，所以精度也不会下降。而基于指纹融合的算法把两种指纹算法融合到一起，提高了精度和召回率。

5　结束语

随着互联网的发展以及机器翻译技术的进步，抄袭剽窃现象越来越严重，而单语言剽窃技术的成熟，促使一些人开始进行跨语言抄袭剽窃，跨语言抄袭剽窃由于存在语言上的不一致以及结构上的不一致，使得跨语言剽窃检测比单语言剽窃检测复杂很多。本文针对此问题提出一种基于指纹融合的跨语言剽窃检测方法。本文主要完成了以下工作：

a）分析了跨语言剽窃检测的主要目的和意义，并详细描述了单语言剽窃检测中的数字指纹技术以及跨语言剽窃检测现阶段国内外所用的主要方法和研究现状，针对跨语言剽窃检测目前理论和方法的不足，提出在 WordNet 上建立中间指纹，运用指纹技术来进行跨语言文本相似度计算。

b）分析了 WordNet 的结构，尤其是关于名词同义词集的树型结构，在深入研究其树型结构的基础上，本文给出了基于其树型结构的中间指纹编码算法，在中间指纹编码算法中子节点以其父节点为前缀，这种方式把语义宽泛与具体以二进制的数字形式表现出来，跨越了语言的障碍。

c）本文重点研究了基于中间指纹的语义消歧和指纹选取。首先分析了目前的文本预处理技术，在此基础上选用适合于本文的预处理技术，包括分词、词性标注等。然后基于中间指纹进行语义消歧，确定特征项在上下文中的唯一义项，接着基于语义频率进行指纹提取工作，过滤掉一些过于宽泛的词所对应的指纹，从而形成文本各自对应的指纹。

d）本文重点研究了基于指纹融合的剽窃检测方法。通过基于中间指纹的跨语言文本段落相似搜索检索出潜在剽窃文档集，然后运用指纹融合算法进行剽窃检测句子级的详细分析[19]。指纹融合是在详细分析 SimHash 算法和 Winnowing 算法之后，在两者基础上提出来的。最后经过剽窃片段合并形成最后的剽窃检测结果，并通过实验验证了该方法的有效性。

跨语言剽窃检测是自然语言处理中的难点之一，本文给出的算法也是对这个问题进行解决的一种方案。但是，由于客观因素的限制，本文所给出的算法中仍有一些问题值得人们来进行完善。第一是各国的 WordNet 中收录的词语并不完全，检测时仍会遇到未登录词。本文所采用的处理方式是将这些词语直接的忽略掉。然而这些被忽略的词语是否会对结果有影响则需要进一步的研究。第二则是由于在 WordNet 当中名词的结构最为明显，所以本文算法只是抽取了文章中的名词进行检测，从而忽略了其他词性的词语对于文章的重要作用，而引入了其他词语后编码问题也将更加的复杂。对于较长的段落来说是合适和高效的，但是对于句子的相似度则不适用，在句子层面还需要经过翻译，而具体的可将疑文档翻译成源文档和将源文档翻译成可疑文档这两个翻译方向是否会对结果造成影响，还需要进一步的研究。

参考文献：

[1] Alzahrani S M, Salim N, Abrabam A. Understanding plagiari-sm linguistic patterns, textual features and detection metho-ds [J]. IEEE Trans on Systems, Man and Cybernetics. 2012, 42 (2): 133-149.

[2] Potthast M, Eiselt A, Barrón-Cedeño A. Overview of the 3rd international competition on plagiarism detection [C]// NotebookPapers of CLEE Labs and Workshop. 2011: 19-22.

[3] Soleman S, Purwarianti A. Experiments on the Indonesian plagiarism detection using latent semantic analysis [C]// Proc of International Conference on Information and Communication Technology. 2014: 5579-5583.

[4] Potthast M, Barrón-Cedeño A, Stein B. Cross-language plagia-rism detection [J]. Language Resources & Evaluation, 2011, 45 (1): 45-62.

[5] Barrón-Cedeño A, Rosso P, Pinto D, Juan A. On cross-lingual plagiarism analysis using a statistical model [C]// Proc of ECAI PAN Workshop. 2008: 9–13.

[6] Ceska Z, Toman M, Jezek K. Multilingual plagiarism detection [C]// Proc of the 13th International Conference on Atifical Inte-lligence: Mtehodology, Systems, and Applications. 2008: 83-92.

[7] Pouliquen B, Steinberger R, Ignat C. Automatic identification of document translations in large multilingual document co-llections [C]// Proc of International Conference on Advances in Natural Language Processing. 2003: 401-408.

[8] Potthash M. Wikipedia in the pocket: indexing technology for near-duplicate detection and high similarity search [C]// Proc of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 2007: 900-909.

[9] Potthast M, Stein B, Barrón-Cedeño A, et al. An evaluate-on framework for plagiarism detection [C]// Proc of the 23rd International Conference on

Computational Linguistics: Posters. 2010: 997–1005.

[10] Pinto D, Civera J, Barrón-Cedeño A, et al. A statistical approach to crosslingual natural language tasks [J]. Journal of Algorithms, 2009, 64 (1): 51–60.

[11] Pereira R C, Moreira V P, Galante R. A new approach for cross-language plagiarism analysis [C]// Proc of International Conference of the Cross-Language Evaluation Forum. 2010: 15-26.

[12] Barrón-Cedeño A, Gupta P, Rosso P. Methods for cross-language plagiarism detection [J]. Knowledge-Based Systems, 2013, 50 (9): 211-217.

[13] Marc F, Parth G, Paolo R. Cross-language plagiarism detection using a multilingual semantic network [C]// Proc of European Conference on Information Retrieval. Berlin: Springer, 2013: 710-714.

[14] Adel A, Masnizah M. Arabic-English cross-language plagiarism detection using winnowing algorithm [J]. Information Technology Journal. 2014, 13 (14): 2349-2355.

[15] Gharghe Z E, Bidgoli B M. Weighted shingling: an adaptation of shingling for weighted shingles [C]// Proc of International Conference on Innovations in Information Technology. 2009: 150-154.